Rendu 3 - NoSQL

1 - Préparation des bancs d'essais

1 - Créer un jeu de test

Nous avons créé avec WatDiv plusieurs jeux de données, un premier contenant 500 000 triplets et un autre en contenant 2 000 000. Le premier nous permettra de mettre en place les tests sans avoir des temps d'exécution trop longs et le deuxième servira à vérifier la solidité des tests.

Nous avons créé aussi une base de requette, contenant 13 000 requettes. Les requettes ont de 1 à 4 conditions, ce qui nous permet de tester sur des requettes plus ou moins complexes.

2 - Histogramme nombre de réponses au requêtes 500K et 2M

Nous avons extrait de ces données le nombre de résultats de chaque requête, avec les 500K et 2M de triplets. Nous visualisons ces données dans ces schémas.

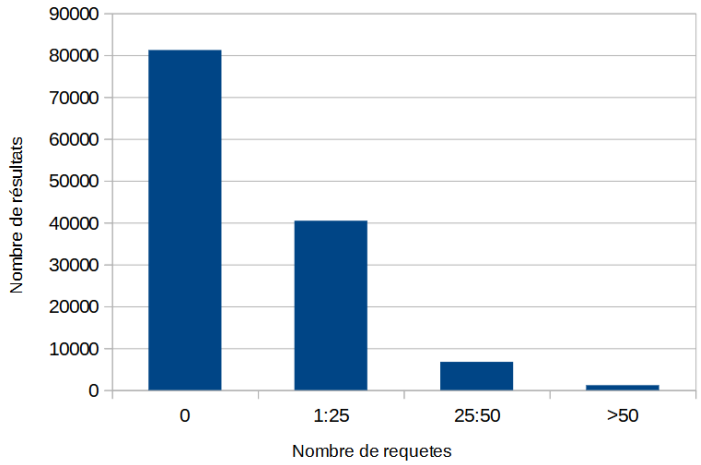

FIGURE 1 – Histogramme avec le jeu de données de 500K triplets

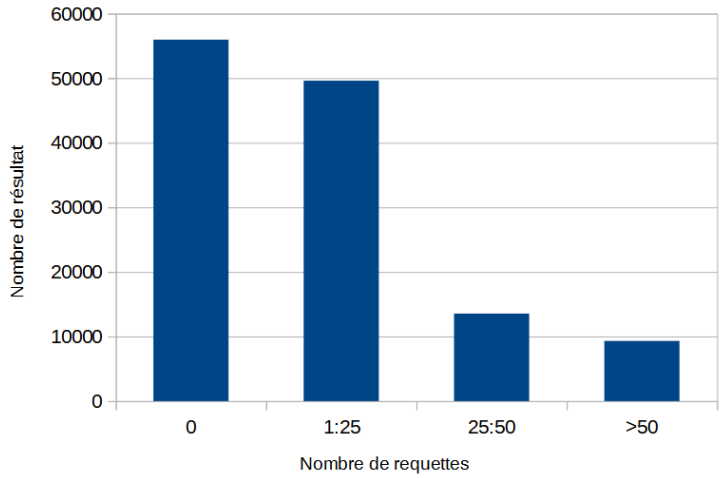

FIGURE 2 – Histogramme avec le jeu de données de 2M triplets

Nous pouvons observer qu'il y a de nombreuses requêtes qui n'ont aucun résultat. Cela semble être les requêtes qui ont beaucoup de conditions et qui sont donc très sélectives. Cependant ce nombre diminue en multipliant le nombre de triplets.

3 - Combien de requêtes ont zéro réponses ? Est-ce souhaitable pour le benchmark ?

Nous avons relevé 5611 requêtes sans résultat sur 13000 requêtes avec le jeu de données contenant 2M de triplets. Nous pensons que ce chiffre est un peu élevé.

4 - Combien de requêtes avec un même nombre de conditions (patrons de requêtes) ? Est-ce souhaitable pour le benchmark ?

Nous avons :

— 1 condition : 5000 requêtes
— 2 condition : 4000 requêtes
— 3 condition : 3000 requêtes
— 4 condition : 1000 requêtes

Pour coller à une utilisation normale d'une base de données, nous pourrions équilibrer ces chiffres pour obtenir quelque chose de plus homogène.

5 - Combien de doublons dans les requêtes ? Est-ce souhaitable pour le benchmark ?

Nous n'avons pas exclu les doublons de notre jeu de requêtes, en effet cela pose problème car ce n'est pas intéressant d'évaluer deux fois la même requête.

6 - Amélioration de notre jeu de requêtes

Nous avons vu que notre jeu de requêtes montrait de nombreux défauts, pour améliorer la qualité de nos tests nous cherchons donc à rendre ce jeu de requêtes le plus représentatif de la réalité.

Nous cherchons donc :

— De 5 à 10% de requêtes sans résultat

— Une répartition du nombre de conditions plus équitables

— Pas de doublons

Notre stratégie pour satisfaire ces contraintes était de générer avec Jéna un fichier Csv contenant chaque requête, le nombre de conditions de la requête et son nombre de résultats.

Ensuite nous utilisons un petit programme en Python pour venir sélectionner dans ce fichier un certain nombre de requêtes. Nous voulons 6000 requêtes, dont 300 avec aucun résultat, 700 avec une condition, 1500 avec 2 ou 3 conditions, et 2000 avec 4 conditions.

2 - Hardware et Software

1 - Quel type de hardware et de software allez vous utiliser pour vos tests ?

Pour effectuer les tests nous allons utiliser la machine personnelle de Matthieu, car cela nous permettra d'avoir le contrôle total de la machine contrairement à celles de la fac où nos permissions sont limitées.

Le programme étant écrit en Java, nous utiliserons la JVM pour l'exécuter, cela nous permet aussi de gérer l'allocation de la mémoire de la machine virtuelle.

La machine embarque un processeur 12 coeurs à 3 Ghz. 8 Gb de mémoire vive, ainsi qu'un SSD.

Les tests seront effectués sur une distribution Linux, à savoir PopOs à la version 22.04. Ce choix vient du fait qu'on a en général plus de contrôle sur une machine Linux

par exemple pour tuer des processus qui pourront réduire les performances durant les tests.

Étant donné que c'est un ordinateur portable, nous effectuons tous les tests avec la machine en charge, pour éviter une bride d'économie de batterie.

Processeur :
— Vendor : AMD
— Model : Ryzen 5 4600H with Radeon Graphics
— Cores : 6 Cores, 12 Threads
— Clockspeed : 3.0GHz
— L2 Cache 3MB
— L3 Cache 8MB
Mémoire vive :
— Size : 8GB RAM
Disque :
— Model : KINGSTON OM8PCP3512F-AB
— Size :476.9 GB
— Sequential Read : 1,358 MBytes/Sec
— Sequential Write : 666 MBytes/Sec
Software :
— OS : PopOs
— Version : 22.04 LTS

2 - Est-ce adapté à l'analyse des performances du système ? Justifiez.

Le matériel n'est pas spécialement inadapté au test car il s'agit d'une machine plutôt puissante et d'actualité. Cependant la machine n'est pas particulièrement adaptée non plus car certains programmes pourraient tourner en arrière-plan, pour cela nous couperons internet, pour éviter les mises à jour en arrière-plan. Notre système n'est pas exactement comme un serveur de base de données car nous n'avons pas la même quantité de RAM ou la puissance de CPU n'est pas la même. Un système plus représentatif serait de tester notre programme directement sur un vrai serveur de base de donnée.

3 - Métriques, Facteurs, et Niveaux

1 - Donnez la liste de métriques permettant d'évaluer les performances de moteurs de requêtes RDF

Nous nous intéresserons à :

— Temps (CPU) d'execution total
— Temps (CPU) de création de l'index
— Temps (CPU) de création du dictionnaire
— Temps (CPU) d'execution d'une requette
— Temps (CPU) d'execution d'un lot de requettes

2 - Listez les facteurs qui rentrent en jeu pendant l'évaluation du système, et définissezen les niveaux

Les éléments qui pourraient venir géner nos test : Nous pouvons imaginer qu'une application, se lance au moment des test pour effectuer une mise a jour ou autre. On pourrai avoir un processus endormi qui prend de la place en mémoire vive, ce qui en laisse moins pour nos tests. On pourrai imaginer que par mégarde on debranche le pc, ce qui entrainerai une chutte de performances.

3 - Ordonnez les facteurs selon leur importance, indiquer les facteurs principaux et secondaires

Ces facteurs du plus important au moins important :

— Le processus qui se lance en arrière-plan.
— La mémoire ram pas entièrement disponible.
— Débrancher l'ordinateur (Pas grave car évitable).